*Trabajo Práctico Conceptos y Paradigmas de lenguaje*

|  |  |
| --- | --- |
|  | |
| ***PARTICIPANTES DEL TRABAJO*** | |
| ***Nombre y Apellido*** | ***DNI*** |
| *Ambesi, Lucas Nahuel* | *39.919.019* |
| *Pereyra, Mauro Lucas* | *38.301.299* |
| *Nativo, Gian* | *32.725.689* |
|  |  |
|  |  |
|  |  |
|  |  |

CONCEPTOS Y PARADIGNMAS DE LENGUAJE

*En este artículo ser hablara de los diferentes Paradigmas inmersos en el mundo de la Programación, en los cuales participan diferentes lenguajes y formas de programación.*

1. *Introducción, ¿Qué es un Paradigma?*

*Podemos hablar de un paradigma como un conjunto de creencias, prácticas y conocimientos que guían el desarrollo de una disciplina durante un período de tiempo. En diversas ramas de la ciencia, un conjunto de ideas en vigencia puede ser reemplazado drásticamente por otro que entre en conflicto con él y se demuestre más acertado.*

*La programación tiene sus propios paradigmas, pero  el término «paradigma de programación» no necesariamente representa un modelo único que deba ser respetado hasta que aparezca otro mejor. De hecho, actualmente muchos paradigmas coexisten en armonía.*

*Un paradigma de programación es un estilo de desarrollo de programas. Es decir, un modelo para resolver problemas computacionales. Los lenguajes de programación, necesariamente, se encuadran en uno o varios paradigmas a la vez a partir del tipo de órdenes que permiten implementar, algo que tiene una relación directa con su sintaxis*

*.*

*Algunos lenguajes de programación pueden soportar****múltiples paradigmas****de programación. Por ejemplo, C++ puede ser empleado para desarrollar software utilizando para ello un modelo de programación puramente orientado a objetos o bien puramente estructurado.*

*Es realmente común el diseño de lenguajes que soporten múltiples paradigmas de programación. Estos lenguajes son aquellos que soportan al menos dos paradigmas. Dentro de esta categoría podemos encontrar nuevas caras y viejos conocidos:*

1. ***Scala****: Imperativo, orientado a objetos, funcional, genérico y concurrente*
2. ***Erlang****: Funcional, concurrente y distribuido*
3. ***Perl****: Imperativo, orientado a objetos y funcional*
4. ***PHP****: Imperativo, orientado a objetos, funcional y reflexivo*
5. ***JavaScript****: Imperativo, orientado a objetos (prototipos) y funcional*
6. ***Java****: Imperativo, orientado a objetos, reflexivo y genérico*
7. ***Python y Ruby****: Imperativo, orientado a objetos, reflexivo y funcional*
8. ***C++****: Imperativo, orientado a objetos, funcional y genérico*
9. ***C#****: Imperativo, orientado a objetos, funcional (lambda), reflexivo y genérico*

*Estos son algunos ejemplos, existen lenguajes como****Oz****que soporta****nueve paradigmas****de programación.*

*Algunos paradigmas*prohíben*el uso de ciertos mecanismos o técnicas. Por ejemplo, en la programación funcional se elimina el uso del*efecto secundario*en las funciones mientras que en la programación estructurada se desaprueba o incluso elimina el uso de la sentencia goto.*

1. *Principales paradigmas de Programación.*

***Imperativo****. Los programas se componen de un conjunto de sentencias que cambian su estado. Son secuencias de comandos que ordenan acciones a la computadora.*

***Declarativo****. Opuesto al imperativo. Los programas describen los resultados esperados sin listar explícitamente los pasos a llevar a cabo para alcanzarlos.*

1. *Paradigma de programación Imperativo*

*En la programación imperativa se describen sentencias que modifican el estado de un programa. En muchos sentidos la programación imperativa es la programación natural para las CPUs que se basan en ese paradigma al nivel**más básico.*

*En este paradigma se expresa como debe solucionarse un problema especificando una secuencia de acciones a realizar a través de uno o más procedimientos denominados subrutinas o funciones.*

*Dentro de esta categoría se engloban la****programación estructurada****, la****programación modular****y la****programación orientada a objetos****. Cada una de estas extensiones o evoluciones han permitido mejorar la mantenibilidad y la calidad de los programas imperativos.*

1. *Programación Estructurada*

*Es una técnica en la cual la estructura de un programa, esto es, la interpelación de sus partes realiza tan claramente cómo es posible mediante el uso de tres estructuras lógicas de control:*

* ***Secuencia****: Sucesión simple de dos o más operaciones.*
* ***Selección****: bifurcación condicional de una o más operaciones.*
* ***Interacción****: Repetición de una operación mientras se cumple una condición.*

*Estos tres tipos de estructuras lógicas de control pueden ser combinados para producir programas que manejen cualquier tarea de procesamiento de información.*

*Un programa estructurado está compuesto de segmentos, los cuales puedan estar constituidos por unas pocas instrucciones o por una página o más de codificación. Cada segmento tiene solamente una entrada y una salida, estos segmentos, asumiendo que no poseen lazos infinitos y no tienen instrucciones que jamás se ejecuten, se denominan programas propios. Cuando varios programas propios se combinan utilizando las tres estructuras básicas de control mencionadas anteriormente, el resultado es también un programa propio.*

*La programación Estructurada está basada en el Teorema de la Estructura, el cual establece que cualquier programa propio (un programa con una entrada y una salida exclusivamente) es equivalente a un programa que contiene solamente las estructuras lógicas mencionadas anteriormente.*

*Una característica importante en un programa estructurado es que puede ser leído en secuencia, desde el comienzo hasta el final sin perder la continuidad de la tarea que cumple el programa, lo contrario de lo que ocurre con otros estilos de programación. Esto es importante debido a que, es mucho más fácil comprender completamente*[*el trabajo*](http://www.monografias.com/trabajos/fintrabajo/fintrabajo.shtml)*que realiza una*[*función*](http://www.monografias.com/trabajos7/mafu/mafu.shtml)*determinada, si todas las instrucciones que influyen en su*[*acción*](http://www.monografias.com/trabajos35/categoria-accion/categoria-accion.shtml)*están físicamente contiguas y encerradas por un bloque. La facilidad de*[*lectura*](http://www.monografias.com/trabajos14/textos-escrit/textos-escrit.shtml)*, de comienzo a fin, es una consecuencia de utilizar solamente tres estructuras de control y de eliminar la instrucción de desvío de flujo de control, excepto en circunstancias muy especiales tales como la*[*simulación*](http://www.monografias.com/trabajos6/sipro/sipro.shtml)*de una estructura lógica de control en un*[*lenguaje*](http://www.monografias.com/trabajos35/concepto-de-lenguaje/concepto-de-lenguaje.shtml)*de programación que no la posea.*

1. ***Ventajas potenciales:***

* *Un programa escrito de acuerdo a estos principios no solamente tendrá una estructura, sino también una excelente presentación.*
* *Un programa escrito de esta forma tiende a ser mucho más fácil de comprender que programas escritos en otros estilos.*
* *La facilidad de comprensión del contenido de un programa puede facilitar el chequeo de la codificación y reducir el*[*tiempo*](http://www.monografias.com/trabajos901/evolucion-historica-concepciones-tiempo/evolucion-historica-concepciones-tiempo.shtml)*de prueba y depuración de programas. Esto último es cierto parcialmente, debido a que la programación estructurada concentra los errores en uno de los factores más generador de fallas en programación: la lógica.*
* *Un programa que es fácil para leer y el cual está compuesto de segmentos bien definidos tiende a ser simple, rápido y menos expuesto a mantenimiento. Estos beneficios derivan en parte del hecho que, aunque el programa tenga una extensión significativa, en*[*documentación*](http://www.monografias.com/trabajos11/ladocont/ladocont.shtml)*tiende siempre a estar al día, esto no suele suceder con los*[*métodos*](http://www.monografias.com/trabajos11/metods/metods.shtml)*convencionales de programación.*
* *La programación estructurada ofrece estos beneficios, pero no se la debe considerar como una panacea ya que el desarrollo de programas es, principalmente, una tarea de dedicación, esfuerzo y*[*creatividad*](http://www.monografias.com/trabajos13/indicrea/indicrea.shtml)*.*

1. ***¿Por qué no se usa GOTO?***

*La programación estructurada como una técnica de programación que no utiliza GOTO.*

*El problema con la instrucción GOTO es que generalmente aleja al programa realizado de los propósitos descritos y en casos extremos puede hacer que un programa sea esencialmente incomprensible.*

1. ***¿Cuáles son los lenguajes de programación Estructurada?***

*Si bien es posible desarrollar la programación estructurada en cualquier lenguaje de programación, resulta más idóneo un lenguaje de programación procedimental. Algunos de los lenguajes utilizados inicialmente para programación estructurada incluyen****ALGOL, Pascal, PL/I y Ada****, pero la mayoría de los nuevos lenguajes de programación procedimentales desde entonces han incluido características para fomentar la programación estructurada.*

1. *Programación Modular*

*La programación modular es un paradigma de programación que consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más legible y manejable.*

*Se presenta históricamente como una evolución de la programación estructurada para solucionar problemas de programación más grandes y complejos de lo que esta puede resolver.*

*Al aplicar la programación modular, un problema complejo debe ser dividido en varios subproblemas más simples, y estos a su vez en otros subproblemas más simples. Esto debe hacerse hasta obtener subproblemas lo suficientemente simples como para poder ser resueltos fácilmente con algún lenguaje de programación. Esta técnica se llama refinamiento sucesivo, divide y vencerás o análisis descendente (Top-Down).*

*Un 'módulo' es cada una de las partes de un programa que resuelve uno de los subproblemas en que se divide el problema complejo original. Cada uno de estos módulos tiene una tarea bien definida y algunos necesitan de otros para poder operar. En caso de que un módulo necesite de otro, puede comunicarse con éste mediante una interfaz de comunicación que también debe estar bien definida.*

*Si bien un módulo puede entenderse como una parte de un programa en cualquiera de sus formas y variados contextos, en la práctica se los suele tomar como sinónimos de procedimientos y funciones. Pero no necesaria ni estrictamente un módulo es una función o un procedimiento, ya que el mismo puede contener muchos de ellos. No debe confundirse el término "módulo" (en el sentido de programación modular) con términos como "función" o "procedimiento", propios del lenguaje que lo soporte.*

1. *Ventajas*

* *Al aplicar la programación modular, un problema complejo debe ser dividido en varios subproblemas más simples, y estos a su vez en otros subproblemas más simples.*
* *En caso de que un módulo necesite de otro, puede comunicarse con éste mediante una interfaz de comunicación que también debe estar bien definida.*
* *Es fácil de mantener y modificar*
* *Es más fácil de escribir y depurar*
* *Facilidad de controlar es decir descompone un problema en estructuras jerárquicas, de modo que se puede considerar cada estructura desde dos puntos de vista.*

1. *Desventajas*

* *No se dispone de algoritmos formales de modularidad, por lo que a veces los programadores no tienen claras las ideas de los módulos*
* *La programación modular requiere más memoria y tiempo de ejecución.*

1. *Programación Procedimental*

*La programación procedimental o programación por procedimientos es un paradigma de la programación muchas veces es aplicable tanto en lenguajes de programación de bajo nivel como en lenguajes de alto nivel. En el caso de que esta técnica se aplique en lenguajes de alto nivel, recibirá el nombre de programación funcional. Esta técnica consiste en basarse de un número muy bajo de expresiones repetidas, englobarlas todas en un procedimiento o función y llamarlo cada vez que tenga que ejecutarse.*

*Esta técnica de programación ofrece muy buena respuesta con relación al tamaño de los programas, y en bruto casi no se nota en la velocidad de ejecución de los mismos (mientras que las variables, constantes o índices de vector estén en memoria, como suele ser normal, estos se relacionarán entre sí, sin una carga de memoria considerablemente alta para los procesadores modernos); aunque es muy complicado conseguir una programación por procedimientos pura.*

*Utilizar este tipo de programación puede resultar muy útil a la hora de programar grandes proyectos, ya que se crea una inmensa biblioteca de funciones especiales para procedimientos utilizados con frecuencia dentro del programa.*

*Por contrapartida, es muy difícil determinar cuál es el número mínimo de instrucciones consecutivas, y el mínimo número que esta secuencia se debe repetir para considerar declarar un procedimiento o una función.*

1. *Lenguajes de Programación*

* *C++*
* [*StarBasic*](https://es.wikipedia.org/wiki/StarBasic)
* [*InfoBasic*](https://es.wikipedia.org/w/index.php?title=InfoBasic&action=edit&redlink=1)
* [*Pascal*](https://es.wikipedia.org/wiki/Pascal_(lenguaje_de_programaci%C3%B3n))
* [*Pascal*](https://es.wikipedia.org/wiki/Pauscal)*en español*

1. *Programación Orientada a Objetos*

*El paradigma orientado a objetos (OO) define los programas en términos de comunidades de objetos. Los objetos con características comunes se agrupan en clases (un concepto similar al de tipo abstracto de dato (TAD)). Los objetos son entidades que combinan un estado (es decir, datos) y un comportamiento (esto es, procedimientos o métodos).*

*Estos objetos se comunican entre ellos para realizar tareas. Es en este modo de ver un programa donde este paradigma difiere del paradigma estructurado, en los que los datos y los métodos están separados y sin relación.*

*El paradigma OO surge para solventar los problemas que planteaban otros paradigmas, como el imperativo, con el objeto de elaborar programas y módulos más fáciles de escribir, mantener y reutilizar.*

1. *Elementos básicos*

* *Bloques:*

*Son un conjunto complejo de datos (atributos) y* [*funciones*](http://www.monografias.com/trabajos7/mafu/mafu.shtml)*(métodos) que poseen una determinada* [*Estructura*](http://www.monografias.com/trabajos15/todorov/todorov.shtml#INTRO)*y forman parte de una organización. Los atributos definen*[*el estado*](http://www.monografias.com/trabajos12/elorigest/elorigest.shtml)*del objeto; los métodos, su*[*comportamiento*](http://www.monografias.com/trabajos16/comportamiento-humano/comportamiento-humano.shtml)*.*

* *Métodos:  
  Es un*[*programa*](http://www.monografias.com/Computacion/Programacion/)*procedimental que está asociado a un objeto determinado y cuya ejecución solo puede desencadenarse a través del mensaje correspondiente.*
* *Mensajes:  
  Es simplemente una petición de un objeto a otro para que este se comporte de una manera determinada, ejecutando uno de sus métodos.  
  Los mensajes comunican a los objetos con otros y con el mundo exterior. A esta técnica de enviar mensajes se la conoce como paso de mensajes.*
* *Clases:  
  Es un tipo definido por el usuario que determina la estructura de datos y las operaciones asociadas con ese tipo.*

1. *Características*

* *Abstracción:  
  Significa extraer las propiedades esenciales de un objeto que lo distinguen de los demás tipos de Objetos y proporciona fronteras conceptuales definidas respecto al punto de vista del observador. Es la capacidad para encapsular y aislar la información de diseño y ejecución.*
* *Encapsulamiento:  
  Es el*[*proceso*](http://www.monografias.com/trabajos14/administ-procesos/administ-procesos.shtml#PROCE)*de almacenar en un mismo compartimiento (una caja negra) los elementos de una abstracción (toda la información relacionada con un objeto) que constituyen su estructura y su  
  Comportamiento. Esta información permanece oculta tanto para los usuarios como para otros objetos y puede ser accedida solo mediante la ejecución de los métodos adecuados.*
* *Herencia:  
  Es la*[*propiedad*](http://www.monografias.com/trabajos16/romano-limitaciones/romano-limitaciones.shtml)*que permite a los objetos construirse a partir de otros objetos.  
  La*[*clase*](http://www.monografias.com/trabajos901/debate-multicultural-etnia-clase-nacion/debate-multicultural-etnia-clase-nacion.shtml)*base contiene todas las características comunes. Las sub-clases contienen las características de la clase base más las características particulares de la sub-clase. Si la sub-clase hereda características de una clase base, se trata de*[*herencia*](http://www.monografias.com/trabajos13/heren/heren.shtml)*simple. Si hereda de dos o más clases base, herencia múltiple.*
* *Polimorfismo:  
  Literalmente significa "cualidad de tener más de una forma". En poo, se refiere al hecho que una misma operación puede tener diferente comportamiento en diferentes objetos. En otras palabras,  
  Diferentes objetos reaccionan al mismo mensaje de modo diferente.*

1. *Ventajas*

* Modelos:

*La poo permite realizar un*[*modelo*](http://www.monografias.com/trabajos/adolmodin/adolmodin.shtml)*de*[*sistema*](http://www.monografias.com/trabajos11/teosis/teosis.shtml)*casi independientemente de los requisitos del Proyecto. La razón es que en la poo la jerarquía la establecen los datos, en*[*cambio*](http://www.monografias.com/trabajos2/mercambiario/mercambiario.shtml)*en la  
Programación estructurada la jerarquía viene definida por los*[*programas*](http://www.monografias.com/Computacion/Programacion/)*.  
Este cambio hace que los*[*modelos*](http://www.monografias.com/trabajos/adolmodin/adolmodin.shtml)*se establezcan de forma similar al razonamiento humano y, por lo tanto, resulte más natural.*

* *Modularidad:  
  Un programa es modular si se compone de módulos independientes y robustos. Esto permite la reutilización y facilita la verificación y depuración de los mismos. En poo, los módulos están  
  Directamente relacionados con los objetos. Los objetos son módulos naturales ya que corresponden a una*[*imagen*](http://www.monografias.com/trabajos7/imco/imco.shtml)*lógica de la realidad.*
* *Extensibilidad:  
  Durante el*[*desarrollo*](http://www.monografias.com/trabajos12/desorgan/desorgan.shtml)*de*[*sistemas*](http://www.monografias.com/trabajos11/teosis/teosis.shtml)*, ocurre la aparición de nuevos requisitos, por eso es deseable que las*[*herramientas*](http://www.monografias.com/trabajos11/contrest/contrest.shtml)*de desarrollo permitan añadirlos sin modificar la estructura básica del diseño. En poo es posible lograr esto siempre y cuando se hayan definido de forma adecuada la  
  Jerarquía de clases, los atributos y métodos.*
* *Eliminación de redundancia en el desarrollo de sistemas se desea evitar la definición múltiple de datos y funciones comunes.  
  En poo esto se logra mediante la herencia (evita la definición múltiple de propiedades comunes a muchos objetos) y el polimorfismo (permite la modificación de métodos heredados). Solo hay que definir los atributos y los métodos en el antepasado más lejano que los comparte.*
* *Reutilización:  
  La poo proporciona un marco perfecto para la reutilización de las clases. El encapsulamiento y la modularidad nos permiten utilizar una y otra vez las mismas clases en aplicaciones distintas. En  
  Efecto, el aislamiento entre distintas clases significa que es posible añadir una nueva clase o un módulo nuevo (extensibilidad) sin afectar al resto de la aplicación.*

1. *Lenguajes de Programación*
   * Puros: Son los que solo permiten realizar programación orientada a objetos.
   * Híbridos: Son los que permiten la poo con la programación estructurada.

* *SmallTalk*
* *C++*
* *Delphi (Object Pascal)*
* *Java*
* *C++*

1. *Paradigma de Programación Declarativa*

*La programación declarativa, en contraposición a la programación imperativa, es un paradigma de programación que está basado en el desarrollo de programas especificando o "declarando" un conjunto de condiciones, proposiciones, afirmaciones, restricciones, ecuaciones o transformaciones que describen el problema y detallan su solución. La solución es obtenida mediante mecanismos internos de control, sin especificar exactamente cómo encontrarla (tan sólo se le indica a la computadora qué es lo que se desea obtener o qué es lo que se está buscando). No existen asignaciones destructivas, y las variables son utilizadas con transparencia referencial. De esta familia hacen parte lenguajes como*Scala*,*Haskell*,*Erlang*y*Elixir.

*Se ha dicho que los lenguajes declarativos tienen la ventaja de ser razonados matemáticamente, lo que permite el uso de mecanismos matemáticos para optimizar el rendimiento de los programas.*

*Son fiables y expresivos.*

*Existen varios tipos de lenguajes declarativos:*

* *Los lenguajes lógicos, como*[*Prolog*](https://es.wikipedia.org/wiki/Prolog)*.*
* *Los lenguajes algebraicos, como*[*Maude*](https://es.wikipedia.org/wiki/Maude_(lenguaje_de_programaci%C3%B3n))*y*[*SQL*](https://es.wikipedia.org/wiki/SQL)*.*
* *Los lenguajes funcionales, como*[*Haskell*](https://es.wikipedia.org/wiki/Haskell)*y*[*Erlang*](https://es.wikipedia.org/wiki/Erlang)

1. *Lenguajes declarativos más conocidos*

* [*Haskell*](https://es.wikipedia.org/wiki/Haskell)*(*[*Programación funcional*](https://es.wikipedia.org/wiki/Programaci%C3%B3n_funcional)*)*
* [*ML*](https://es.wikipedia.org/wiki/ML_(lenguaje_de_programaci%C3%B3n))*(*[*Programación funcional*](https://es.wikipedia.org/wiki/Programaci%C3%B3n_funcional)*)*
* [*Lisp*](https://es.wikipedia.org/wiki/Lisp)*(*[*Programación funcional*](https://es.wikipedia.org/wiki/Programaci%C3%B3n_funcional)*)*
* [*Prolog*](https://es.wikipedia.org/wiki/Prolog)*(*[*Programación lógica*](https://es.wikipedia.org/wiki/Programaci%C3%B3n_l%C3%B3gica)*)*
* *F-Prolog (Programación lógica difusa)*
* [*Curry*](https://es.wikipedia.org/wiki/Curry_(lenguaje_de_programaci%C3%B3n))*(Programación lógico-funcional)*
* [*SQL*](https://es.wikipedia.org/wiki/SQL)
* [*QML*](https://es.wikipedia.org/wiki/QML)

1. *Paradigma Funcional*

*En ciencias de la computación, la programación funcional es un paradigma de programación declarativa basado en el uso de funciones matemáticas, en contraste con la programación imperativa, que enfatiza los cambios de estado mediante la mutación de variables. La programación funcional tiene sus raíces en el cálculo lambda, un sistema formal desarrollado en los años 1930 para investigar la definición de función, la aplicación de las funciones y la recursión. Muchos lenguajes de programación funcionales pueden ser vistos como elaboraciones del cálculo lambda.*

*En la práctica, la diferencia entre una función matemática y la noción de una "función" utilizada en la programación imperativa, es que las funciones imperativas pueden tener efectos secundarios, como cambiar el valor de cálculos realizados previamente. Por esta razón carecen de transparencia referencial, es decir, la misma expresión sintáctica puede resultar en valores diferentes en varios momentos de la ejecución del programa. Con código funcional, en contraste, el valor generado por una función depende exclusivamente de los argumentos alimentados a la función. Al eliminar los efectos secundarios se puede entender y predecir el comportamiento de un programa mucho más fácilmente. Ésta es una de las principales motivaciones para utilizar la programación funcional.*

* *Función Pura:*

*Las funciones puramente funcionales (o expresiones) no tienen efectos secundarios (memoria o E/S).*

* *Función Orden Superior:*

*Funciones de orden superior son funciones que pueden tomar otras funciones como argumentos o devolverlos como resultados. En cálculo, un ejemplo de una función de orden superior es el operador diferencial d / dx, que devuelve la derivada de una función f.*

* *Recursividad:*

*Iterar en los lenguajes funcionales es normalmente llevado a cabo mediante recursividad. Las funciones recursivas se invocan a sí mismas, permitiendo que una operación se realice una y otra vez hasta alcanzar el caso base.*

1. *Ventajas*

* *Ausencia de efectos colaterales*
* *Proceso de depuración menos problemático*
* *Pruebas de unidades más confiables*
* *Mayor facilidad para la ejecución concurrente*

1. *Lenguajes de programación Funcional*

* *Lenguajes Puros:*
* *Haskell*
* *Miranda*
* *Lenguajes Funcionales:*
* *Scala*
* *Lisp*
* *Clojure*
* *Scheme*
* *Ocaml*
* *SAP*
* *Standard ML*
* *Funcional de Programación Concurrente:*
* *Erlang*

*Los lenguajes de programación funcional, especialmente los puramente funcionales, han sido enfatizados en el ambiente académico y no tanto en el desarrollo comercial o industrial. Sin embargo, lenguajes de programación funcional como Scheme, Erlang, Rust, Objective Caml, Scala, F# y Haskell, han sido utilizados en aplicaciones comerciales e industriales por muchas organizaciones. La programación funcional también es utilizada en la industria a través de lenguajes de dominio específico como R (estadística), Matemática (matemáticas simbólicas), J y K (análisis financiero).*

*Los lenguajes de uso específico usados comúnmente como SQL y Lex/Yacc, utilizan algunos elementos de programación funcional, especialmente al procesar valores mutables. Las hojas de cálculo también pueden ser consideradas lenguajes de programación funcional.*

*La programación funcional también puede ser desarrollada en lenguajes que no están diseñados específicamente para la programación funcional. En el caso de Perl, por ejemplo, que es un lenguaje de programación imperativo, existe un libro que describe cómo aplicar conceptos de programación funcional. JavaScript, uno de los lenguajes más ampliamente utilizados en la actualidad, también incorpora capacidades de programación funcional. Python también incorpora particularidades de los lenguajes funcionales como listas de comprensión y funciones de tratamiento de listas como matemática de conjuntos. Java en su versión 8, está incorporando la programación funcional, así como el uso de las expresiones lambda.*

1. Paradigma Lógico

*Paradigma de programación basado en la lógica de primer orden. La Programación Lógica estudia el uso de la lógica para el planteamiento de problemas y el control sobre las reglas de inferencia para alcanzar la solución automática.*

*Se trabaja en una forma descriptiva, estableciendo relaciones entre entidades, indicando no como, sino que hacer, entonces se dice que la idea esencial de la Programación Lógica es:*

*PROGRAMA = LOGICA (determina las soluciones producidas) + CONTROL (formas alternativas de ejecutar Lógica).*

*Se puede ver como una deducción controlada.*

***Lógica (programador):****hechos y reglas para representar conocimiento.*

***Control (interprete):****deducción lógica para dar respuestas (soluciones).*

1. *¿Que intenta resolver?*

*Dado un problema S, saber si la afirmación A es solución o no del problema o en qué casos lo es. Además queremos que los métodos sean implantados en máquinas de forma que la resolución del problema se haga de forma automática*

***La programación lógica: construye base de conocimientos mediante reglas y hechos.***

* *Hechos*

*Declaración, cláusula o proposición cierta o falsa, el hecho establece una relación entre objetos.*

*Son las sentencias más sencillas. Un hecho es una fórmula atómica o átomo: p(t1,...,tn)p(t1,...,tn) e indica que se verifica la relación (predicado)****p****sobre los objetos (términos) t1,...,tnt1,...,tn.*

* *Reglas*

*Implicación o inferencia lógica que deduce nuevo conocimiento, la regla permite definir nuevas relaciones a partir de otras ya existentes*

* *Consultas*

*Se especifica el problema, la proposición a demostrar o el objetivo*

1. *Características*

* *Unificación de términos*
* *Mecanismos de inferencia automática*
* *Recursión como estructura de control básica*
* *Visión lógica de la computación*

1. *Ventajas*

* *Puede mejorarse la eficiencia modificando el componente de control sin tener que modificar la lógica del algoritmo.*
* *Relaciones multipropósito.*
* *Simplicidad.*
* *Generación rápida de prototipos e ideas complejas.*
* *Sencillez en la implementación de estructuras complejas.*
* *Potencia.*

1. *Desventajas*

* *Altamente ineficiente.*
* *Pocas áreas de aplicación*
* *No existen herramientas de depuración efectivas.*
* *En problemas reales, es poco utilizado.*
* *Si el programa no contiene suficiente información para contestar una consulta responde false.*

1. *Lenguajes de Programación*

* *Prolog*
* *Mercury*
* *CLP (FD)*
* *CSP (Constraint Satisfaction Problem)*
* *Lambda Prolog*
* *Logtalk*
* *Alma-0*
* *CLAC(Logical Composition with the Assistance of Computers)*
* *Gödel*
* *Curry*
* *Ace*
* *PALs*
* *Actor Prolog*

1. *Prolog*

*Es un Lenguaje de Programación diseñado para representar y utilizar el conocimiento que se tiene sobre un determinado dominio. Los programas en Prolog responden preguntas sobre el tema del cual tienes conocimiento*

*.La popularidad del lenguaje se debe a su capacidad de deducción y además es un lenguaje fácil de usar por su semántica y sintaxis. Solo busca relaciones entre los objetos creados, las variables y las listas, que son su estructura básica.*

*Escribir un programa en Prolog consiste en declarar el conocimiento disponible acerca de objetos, además de sus relaciones y sus reglas, en lugar de correr un programa para obtener una solución, se hace una pregunta, el programa revisa la base de datos para encontrar la solución a la pregunta, si existe más de una solución, Prolog hace un barrido para encontrar soluciones distintas. El propio sistema es el que deduce las respuestas a las preguntas que se le plantean, dichas respuestas las deduce del conocimiento obtenido por el conjunto de reglas dadas.*

1. *CONCLUCION*

*En base a este documento y las prácticas en los diferentes lenguajes aprendimos que hay más de una forma de programar, el gran asombro se dio asociado al Paradigma Declarativo del cual desconocíamos su funcionamiento de “Caja Negra”, o la utilización de lenguajes solo asociados a la Matemática.*

*Este trabajo sirvió para agrandar nuestro abanico de opciones a la hora de encarar un nuevo Proyecto, y no quedarnos encerrados en una sola manera de resolverlo.*